Early Detection of DDoS Attacks in SDN using Machine Learning Models

Authors

Refah Rafia Islam

Fahim Mahmood

Tabia Mosharref

Supervisor

Dr. Md. Moniruzzaman

Assistant Professor,Department of CSE

Islamic University of Technology (IUT)

Co-Supervisor

Faisal Hussain

Lecturer,Department of CSE

Islamic University of Technology (IUT)

A thesis submitted in partial fulfilment of the requirements for the degree of B. Sc. Engineering in Computer Science and Engineering

Academic Year: 2020-2021

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

A Subsidiary Organ of the Organization of Islamic Cooperation (OIC)

Dhaka, Bangladesh

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Tabia Mosharref, Fahim Mahmood and Refah Rafia Islam under the supervision of Dr. Md. Moniruzzaman, Assistant Professor of the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Refah

Refah Rafia Islam

Student ID - 170041051

[illegible]

Fahim Mahmood

Student ID - 170041027

[illegible]

Tabia Mosharref

Student ID - 170041010

Approved By

Supervisor

[illegible]

Dr. Md. Moniruzzaman

Assistant Professor, Department of CSE

Islamic University of Technology (IUT)

Co-Supervisor

[illegible]

Faisal Hussain

Lecturer, Department of CSE

Islamic University of Technology (IUT)

Acknowledgement

We would like to express our grateful appreciation for **Assistant Professor Md. Moniruzzaman**, Department of Computer Science & Engineering, IUT for being our adviser and mentor. His motivation, suggestions and insights for this research have been invaluable. Without his support and proper guidance this research would never have been possible. His valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithm, modification till the project implementation and finalization which helped us to do our thesis work in proper way. We are really grateful to him.

We are also grateful to **Faisal Hussain**, Lecturer, Department of Computer Science & Engineering, IUT for his valuable inspection and suggestions on our proposal of DDoS Attack Detection scheme in SDN network using Machine Learning models.

Abstract

Software Defined Networks (SDN) are programmable networks that can be easily managed with a global understanding of network topology. However, while the software-defined network architecture enhances network resource pooling by separating the control layer from the data layer, this centralized management and control introduces security vulnerabilities into the SDN architecture. One of the most dangerous attacks that the SDN architecture faces is distributed denial of service (DDoS). Aiming at the detection of DDoS attacks under the SDN architecture, this paper proposes faster DDoS attack detection using machine learning based classifier XGBoost which provides higher accuracy.

Keywords: *SDN, XGBoost, Random Forest, Decision Tree, KNN, SVM, CICDDoS 2019, DDoS*

Contents

List of Figures

List of Tables

1 Introduction

A detailed background of history of Software Defined Networking (SDN), our problem statement and objectives are discussed in details in this section.

1.1 Overview

Software Defined Network (SDN) is often defined with terms like mininet, openflow, Ryu, ODL, and OVS. However, this is only partially correct. SDN is essentially composed of all of these components. Numerous network devices such as switches, routers, firewalls, IDS(Intrusion Detection System), IPS(Intrusion Prevention System), and load balancers are available. All of these devices are composed of two components: the data plane, which transmits packets, and the control plane, which determines how the data plane should act and how packets should be forwarded. These traditional network devices have a number of drawbacks. They are vendor-specific, with hardware and software combined. Costs vary according to technology but are extremely high. Additional features can be added only at the vendor's discretion. The client can only request features; the vendor will decide whether to add them or not, and the vendor will choose the time frame in which these features will become accessible. The devices are purpose-built. We cannot configure a router to act as a load balancer, or a switch to act as a firewall, or vice versa.If a network contains hundreds of these devices, each one must be set separately. There is no centralized administration.

Unlike traditional networks, where the forwarding plane and data plane are tightly coupled, SDN networks have decoupled both the hardware and software components, such that the control plane is distinct from the forwarding plane, and the planes communicate using a protocol called OpenFlow. The OpenFlow protocol, Openflow-based switches, SDN controllers (POX, Ryu, OpenDayLight, Floodlight), and SDN applications are the primary components of the SDN architecture (hub, switch, router, firewall, load balancer). The control plane communicates with the simple networking devices, i.e. the SDN controller advises the switches via the southbound openflow protocol what actions they should do. Northbound openflow protocol is used to communicate between the control plane and the application layer. A simple SDN architecture is given below:

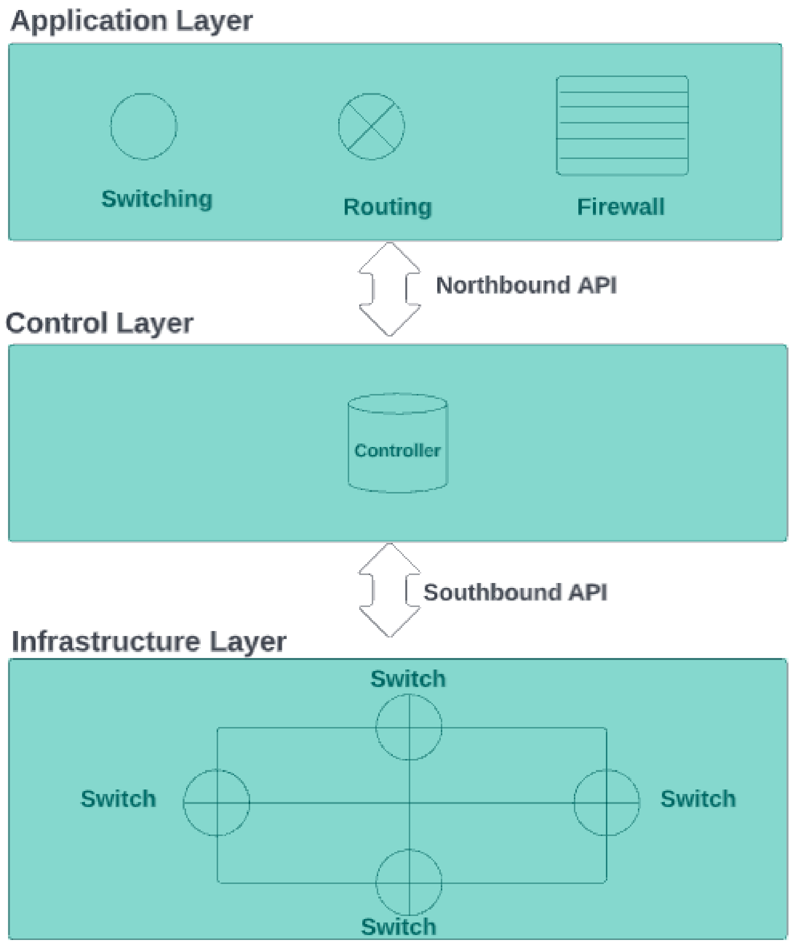


Figure 1: SDN Architecture

When a packet arrives at the switch, one of two things can happen: 1. packet that matches a flow entry (carries out the actions described in the flow entry); 2. packet that does not match any of the flow entries (forwarded to the controller as Packet In message). The OpenFlow switch maintains flow tables that are used to perform packet lookups and routing choices. The flow table is the most critical component of OpenFlow switches. Each packet that enters the switch must travel through one or more flow tables. Flow refers to the sequencing of packets. The OpenFlow protocol defines the messages that are sent between an OpenFlow controller and an OpenFlow switch. Three distinct sorts of messages exist:

• The controller starts messages between the control plane and the switch in order to control or view the status of the switch directly.

• Symmetric communications are transmitted in both ways without being solicited.

• Asynchronous messages are transmitted from the controller to the switch without the controller requesting them.

Because traditional networks can't handle today's dynamic, scalable processing and storage needs, SDN has emerged as a new paradigm for network design. High-bandwidth dynamic applications are perfectly suited to SDN's dynamic, controlled, cost-effective, and adaptive design. Using the controller as an operating system, real-time installations and modifications may be performed. SDN's unique capabilities allow it to be used in a wide range of network scenarios, from home and business networks to data centers and cloud networks. The centralized control plane in software-defined networking (SDN) simplifies network management, but it also creates several potential security risks. A topology of SDN is given below:

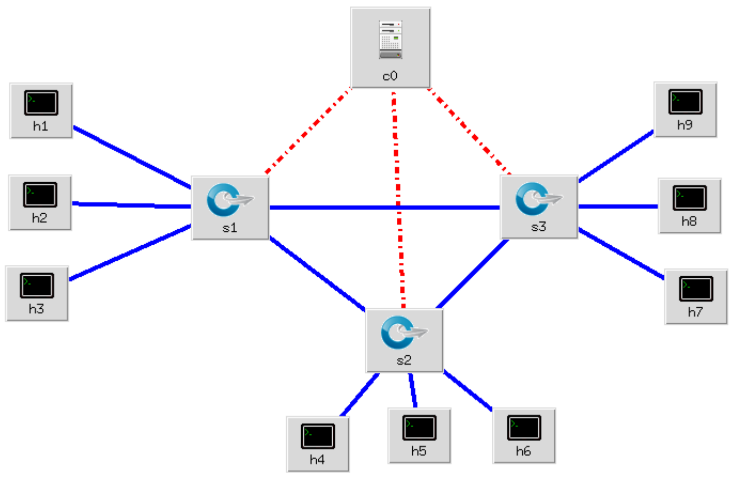


Figure 2: SDN Topology

1.2 Thesis Objectives

SDN has improved the world of networking by giving users more flexibility and convenience in resource utilization, hence easing network administration. Multiple methods exist for mitigating the vulnerabilities that SDN has introduced. Several papers have analyzed the risks and threats posed by SDN, as well as detection techniques for mitigating these dangers. Given the significance of SDN technology in networking, our primary objective has been to bring improvement in the detection mechanism of SDN risks.

1.3 Problem Statement

SDN (software-defined networking) is a networking technology that manages networks by allowing network configuration to be programmed efficiently in order to increase the performance of the network. By creating a physical separation between the network packets sent by the forwarding process and the SDN control layer sent by the routing process, SDN can construct a centralized intelligence for a single network component. The control plane serves as the network's brain. [4] A number of disadvantages of centralized designs exist, including security, elasticity, and scalability issues, to name a few. As a result of the architecture's centralized structure, security is the primary concern of SDN users and administrators. DDoS attacks are a huge danger to the security of the Internet. The purpose of this attack is to render a targeted service unavailable by flooding the internet with traffic. By depleting the network's resources, delays and interruptions are caused.

Different sorts of DDoS attacks have been documented in recent years. A DDoS attack renders the resource unavailable and limits service availability for a certain period of time, resulting in service failures, resource exhaustion and increased expenses. As a result, detecting DDoS attacks has become a significant research topic. Numerous studies in this area have been conducted till now, including statistical methodologies, entropy-based approaches, and machine learning models.

So far, all research has been conducted using datasets of traditional networks.

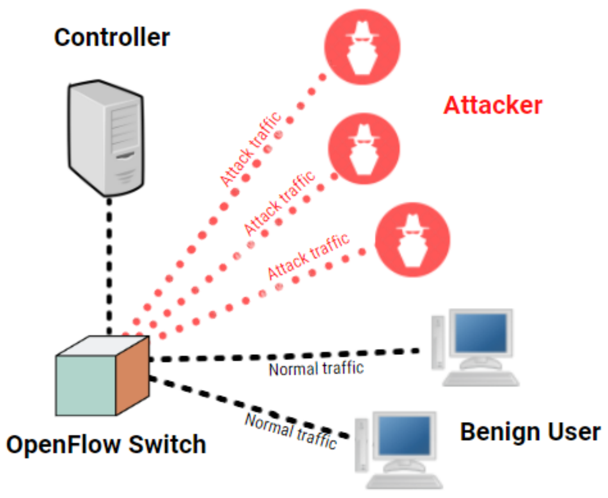


Figure 3: DDoS Attack

Most of the advanced machine learning-based solutions have used outdated benchmark datasets instead of SDN-specific datasets for training purposes. As a result, we propose this work, in which we have used an SDN-specific dataset to get a more practical model that will work in a real SDN environment. Additionally, what makes this research topic more significant is the requirement for early detection of such denial of service attacks with reduced execution time, in addition to accuracy. As a result, keeping both aspects in consideration, we provide a solution that gives highest accuracy of detection with lowest execution time.

1.4 Thesis Contribution

The contribution of this paper is to propose such a detection method that gives the minimum execution time and at the same time highest accuracy. We compared the most prominent methods, capable of detecting SDN DDoS attacks, in terms of their accuracy, false alarm rate and execution time, along with our proposed methodology in order to demonstrate how our classifier provides a significant improvement.

1.5 Research Challenges

One major challenge in our research scope is the fact that DDoS attacks indicate an increasing attack scope; the attack mode is also increasingly intelligent. The various challenges in detecting DDoS attacks are due as follows :

1. the attack traffic characteristics are difficult to identify
2. there is a lack of collaboration between coherent network nodes
3. the attack tool is strengthened, while the threshold of its use decreases
4. widely used address fraud makes tracing the source of the attack difficult
5. the attack duration time is short and response time is limited.

Security concerns are expected to grow as SDN technologies are gradually deployed.

1.6 Organization of Thesis

The rest of the thesis is organized in the following manners. Section 2 canvasses prior works related to our topic of interest. Section 3 discusses our proposed detection approach. Section 4 discusses how we conducted our experiment and set up the environment required for it. Section 5 discusses result analysis and hence performance evaluation. Section 6 mentions the future work in mind with regard to our research topic. Finally, section 7 draws the conclusion.

2 Literature Review

Many authors have proposed DDoS attack detection and mitigation using different techniques such as evolutionary algorithms, genetic algorithm, machine learning techniques, swarm particle optimization, fuzzy logic etc. We mainly researched and analysed DDoS detection technique in SDN using Entropy and machine learning techniques and mentioned our findings in this chapter.

2.1 DDoS Detection using Entropy

Mousavi et al. [5] proposed a methodology for recognizing DDoS attacks based on the destination IP address's entropy fluctuation. Their approach quantifies the unpredictability of incoming packets using entropy. Each host should have a high likelihood of receiving fresh incoming packets in a 64-host network. As a consequence, the degree of entropy is increased. When one or more hosts experience an exceptionally high volume of incoming packets, randomness and entropy degrade. The entropy attribute is used in this study to detect an attack in its early phases. They established an experimental threshold for entropy in the research based on the simulation results, and values less than the threshold are deemed attacks.

To identify DDoS attacks using entropy, two critical components are required: i) a window size and ii) a threshold. The window's size is determined by the time period or the amount of packets. Within this duration, entropy is calculated to determine the degree of uncertainty in the incoming packets. If n is the number of packets included inside a window and pi is the probability of each element being within the window, the entropy (H) is calculated as follows:

$$H = -\sum_{i=1}^{n} p_i \log p_i$$

Entropy is calculated only by the destination IP address for newly incoming packets. Entropy falls during an attack, as the IP address of the attacked host(s) appears more often. The window size is set to 50 due to the network's restricted capacity for new connections. Unrouted packets are retained forever in the absence of a fresh request. A controller is restricted in its ability to link switches and hosts. Its little size significantly minimizes the amount of processing required for each window. A 50-packet attack is more quickly detected than a 500-packet attack. Memory use remains constant, while CPU consumption increases somewhat.

The destination IP address of incoming packets is monitored in order to determine whether the controller is under attack. Entropy is maximized when each IP address occurs just once. When a host is attacked, it receives a huge quantity of packets. These packets will fill the window, reducing the number of unique IP addresses and so increasing entropy. As a consequence, an experimental cutoff point is determined. If the entropy goes below this value for five consecutive windows, an attack is conducted. The attack detects 250 packets and informs the network in five entropy intervals. The proposed technique may be summarized in four parts:

- **Using SDN Capabilities:** With each new incoming connection, the controller creates a flow in the switch, which routes the remaining incoming packets directly to the destination. As a result, each time the controller encounters a packet, it treats it as fresh. Another well-known characteristic about newly received packets is that the destination host is inside the controller's network. This assumption is made on the basis of an attack on one of the network's hosts or a subnet of network hosts. The network is made up of switches and hosts. With the packet's origin and destination known, the quantity of randomness may be determined by computing the entropy for a given window size. Maximum entropy is achieved when each packet is destined for a single host. On the other hand, when all packets in a window are meant for a single host, the lowest entropy is achieved. For example, if a window comprises 64 components and each component appears only once, the entropy value is 1.80. When a single element appears ten times, the entropy is 1.64. This entropy property will be utilized to compute the randomization of SDN controllers. We can examine the rate at which new packets arrive to the controller and establish whether or not an attack is occurring due to the controller's centralized view of the network.

Due to its ability to quantify randomness and to calculate a minimum and maximum entropy value based on entropy, this becomes a viable solution for DDoS detection. When a high volume of packets arrives at a single host or a subnet of hosts, the entropy value decreases.

Collecting entropy statistics: Collecting entropy information from switch tables is one of the controller's duties. After an inert time, the controller evaluates existing flows and eliminates inactive ones. This functionality is used in this study to enhance the controller's data collection capabilities. They provided the source IP address of newly received packets, which were to be clustered into 50-byte windows. The entropy of each window is computed and compared to a specified experimental value. An assault is detected when the entropy value goes below a predefined threshold. Due to these two characteristics, the approach is adaptable to a wide variety of controllers.

Window Size: They have picked a 50-inch window. The primary rationale for selecting 50 is the limited amount of new connections that each server in the network may accept. Once a connection is established, packets are not routed via the controller until a new request is received. A second problem is that each controller is limited in terms of the number of switches and hosts it can connect.Finally, selecting this size minimizes the amount of processing required for each window. As the size of the window increases, so does the CPU utilization. Given the controller's limited resources, this window size is optimal for networks consisting of a single controller and a few hundred hosts.

Attack Detection: The destination IP address of incoming packets is monitored to determine whether the controller is under attack. When an IP address is added to the database for the first time, it is assigned a count of one. If it appears in the table, its count will be increased. The window's entropy will be determined after 50 packets. Entropy is maximized when each IP address occurs just once. When a host is attacked, it receives a tremendous volume of packets. These packets will use the majority of the window's available space, reducing the number of unique IP addresses and so entropy. If the entropy value falls below this level and five successive windows have entropy values less than the threshold, an attack is underway. Within five entropy periods, 250 packets in the attack are detected, alerting the network to the attack.

Problem: Although using entropy seemed to be a viable solution, it has several limitations: it could detect when only a single host is under attack and specific to only one type of DDoS attack. The new novel DDoS attack surpass without being detected by the entropy approach.

2.2 SDN Dataset

Most of the SDN DDoS detection systems using different machine learning approaches use data generated from conventional networks. To the best of our knowledge, there is no well-established benchmark/standard dataset for SDN DDoS detection. Some common benchmark dataset used in IDS from conventional networks are mentioned below:

2.2.1 KDD'99

This is one of the most well-known datasets widely used in IDS training. It contains 41 features and contains four types of attacks: Denial of Service (DoS), Remote to Local (R2L), User to Root (R2L) and Probe attack. This dataset has a lot of duplicate entries and is biased toward DoS attacks.

2.2.2 NSL-KDD

This is an updated version of KDD'99 dataset. It has additional 17 attacks and uses an outdated version of TCP protocol. Even though this is a modified version of KDD'99, this dataset can't reflect the current attack trends as this was captured almost 20 years ago. Only 6 out 41 features concern SDN according to literature.

2.2.3 Kyoto Dataset

This dataset was collected from servers in Kyoto University from 2006-2009. Though the dataset contains real traffic, the attack types are unknown and have an imbalanced class distribution making it a biased dataset. We haven't come across SDN DDoS detection models trained using this dataset yet.

2.2.4 CICIDS 2017

This is one of the most recent dataset with 80 features covering comprehensive attack scenarios. This was released based on the foundation of ISCX2012 dataset published in 2012. It contains many redundant records and almost 40 features are irrelevant to SDN DDoS detection. Recently, CICIDS 2019 has been published which is an updated version of CICIDS 2017.

2.2.5 InSDN Dataset

As majority of the literature use incompatible and outdated dataset for anomaly detection in SDN, InSDN [3] was proposed in 2020 as attack-specific SDN dataset. This dataset has 7 attack classes generated in a virtual environment: DoS, DDoS, Web attacks, R2L, Malware, Probe attack and U2R. The dataset was generated from low scale topology. The authors evaluate their dataset on 8 machine learning algorithms: KNN, Decision Tree, Random Forest, AdaBoost, Naive Bayes, rbf-SVM, lin-SVM and MLP.

2.3 DDoS Detection Machine Learning Technique

SDN DDoS detection using different machine learning techniques have gained popularity recently and many literature have done comparative analysis on them. But the dataset used were from traditional and conventional networks. In 2018, a new type of DDoS was proposed by Di Wu et al. [8]. As far as we are aware, there is no standard SDN DDoS dataset other than InSDN and let alone any benchmark dataset which takes this new type of DDoS into consideration. Traditional DDoS used to target a single victim using multiple source ip to mimic the transmission from different sources. New type of DDoS attack targets multiple victims/destinations (even one's that are not in the network). Traditional DDoS takes down the switch by sending huge amounts of packets to one victim whereas the new type of DDoS attack has the potential to take down the whole network by targeting the controllers. We briefly discuss the common supervised machine learning techniques to detect DDoS in SDN:

2.3.1 SVM

Jin Yu et al. [12] proposed that existing neural network algorithms are not practical solutions to be applied in SDN for anomaly detection. The author proposed SVM classification algorithm to detect DDoS in SDN and obtained 95.24% average accuracy. The authors didn't evaluate their accuracy on any standard dataset. They generated their own flow information and extracted 6 tuple characteristics:

- Speed of Source IP (SSIP)
- Speed of Source Port (SSP)
- Standard Deviation of Flow Packets (SDFP)
- Deviation of Flow Bytes (SDFB)
- Speed of Flow Entries (SFE)
- Ration of Pair-Flow (RPF)

DDoS attacks are executed by sending UDP/IMCP/SYNC packets. ICMP traffic doesn't source port and destination port. So, the SSP and RPF become zero making 6-tuple characteristics to 4-tuple characteristics. Daniel Burgos et al. [7] in 2020 proposed SVM with KPCA (Kernel Principal Component Analysis) with Genetic algorithm (GA). KPCA is used for feature reduction and GA is used to find optimal SVM parameters. The author obtained 98.907% accuracy on the NSL-KDD dataset. One of the biggest drawbacks of SVM is that it takes a lot of time to train.

2.3.2 Random Forest(RF)

This paper [9] uses a combination of trigger module based on Gini impurity and Random Forest classifier to detect DDoS in a SDN environment. Controller requests the switch for flow information. After receiving the flows, the controller doesn't execute the random forest classifier. Instead, gene impurity is calculated using two features (source IP destination IP). If the gini impurity is greater than a certain threshold, the random forest classifier is triggered. The random forest classifier is trained on ISCX-DDoS 2012 dataset. The author also compares Random Forest with SVM and Decision Tree. The detection rate of Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT) are 98.63%, 97.1% and 97.86% respectively.

2.3.3 Decision Tree (DT)

The authors of [4] propose detection of DDoS using Decision Tree, Naive Bayes and SVM. They evaluated their technique using a custom dataset generated from linear SDN topology with 3 switches and 6 hosts in a virtual environment, CIC-DDoS 2019 dataset and hybrid dataset (custom + CIC-DDoS 2019). Decision tree gives the best result in all 3 types of dataset and the average accuracy is 99.9%.

2.3.4 XGBoost

Zhuo Chen et al. [1] extreme gradient boosting algorithm using JDD'99 dataset. The authors use 9 features out of 41 from KDD '99 and obtained accuracy of 98.53%.

2.3.5 KNN

Huseyin Polat et al. [6] mentioned that use of wrapper feature selection with KNN classifier achieved 98.3% accuracy in DDoS detection of SDN. The 12 features used in training are:

- Ifinpkt
- Ifoutpkt
- Hits
- Miss
- OpenFlowforum
- Cpu-util
- Mngmnt_interface_inpkts
- Mngmnt_interface_outpkts
- OpenFlowto
- OVSFlow
- OVSLost

Liang Tan et el. [10] proposed a new framework for detecting DDoS in SDN. The framework consists of K-Means based training data processing module and K-nearest neighbor (KNN) based traffic detection module. Average precision of KNN algorithm and K-Means algorithm are 95.83% and 95.99% respectively. Average precision of the framework proposed by the author is 99.03%.

2.4 Novel DDoS

In [11], Di Wu et al. revealed a new sort of DDoS attack that targets the SDN environment and is more difficult to detect. Authors suggested a unique real-time DDoS detection system for new type of DDoS by analyzing network status data via Principal Component Analysis (PCA), statistical trapezoidal model in the literature.

2.4.1 Principal Component Analysis (PCA)

As described in [11], this new sort of DDoS attack is distinct from traditional DDoS attacks in that the packet's destination is chosen at random. This attack is not directed against a specific server, but at the SDN network system as a whole. Thus, there will be no server that detects an attack, and thus no server that will warn the attack, making it more difficult to identify and report. By comparing with a sample entropy, they identified that when the destination IP becomes random as in the new DDoS attack, sample entropy cannot detect it but PCA can.

2.4.2 Statistical Trapezoidal Model

Bakhtiari et al in [8] introduced a new linear regression based method that can detect the recent DDoS attack which entropy-based methods are incapable of detecting. This paper used a statistical trapezoid model to estimate the amount of table misses for each switch based on the SDN structure and traffic analysis. Linear regression and EWMA estimates are then used to estimate the table's threshold misses at defined time intervals. Using this strategy, it has been said that one can detect DDoS attacks in software defined networks at an early stage, independent of the type of DDoS attack. The suggested approach for detecting denial of service attacks is divided into three general stages: fetching, estimation, and detection.

The controller first retrieves the number of received packets and table misses for each switch per second. Second, the equations of threshold lines are estimated at t seconds. The intercept parameters and slope for the following interval are computed using EWMA. Third, the number of table misses is compared with the estimated value of the threshold. If the threshold value is surpassed, a DDoS attack is detected. Additionally, the paper says that entropy and PCA approaches cannot completely detect this form of attack.

3 Proposed Approach

In order to address the issue of accuracy, execution speed and false alarm rate , we propose an extreme gradient boosting algorithm i.e XGBoost algorithm classifier as the detection algorithm. XGBoost classifier is a distributed gradient boosted tree classifier that aggregates hundreds of tree models of lower accuracy and generates a model with higher accuracy and lower false positive rate.

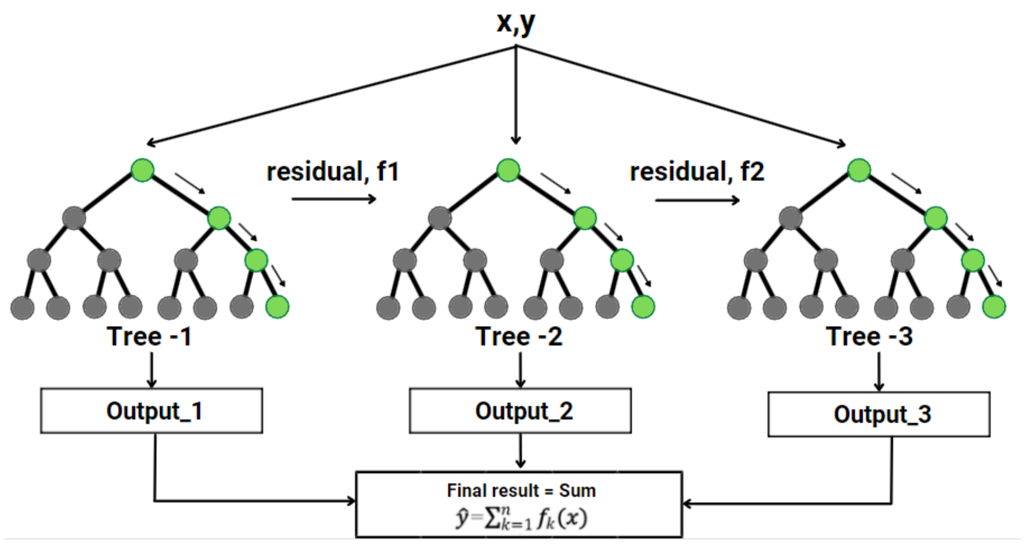


Figure 4: Simplified XGBoost Architechture

3.1 XGBoost Classification Algorithm

In this part we discuss the working method XGBoost algorithm. XGBoost is an efficient and improved implementation of the traditional Gradient Boosted Decision Tree algorithm in respect to computational speed, performance and scalability . The objective function of XGBoost is as follow:

$$F_{Obj}(\theta) = L(\theta) + \Omega(\theta)$$

$$where L(\theta) = l(\hat{y_i}, y_i) \tag{1}$$

$$\Omega(\theta) = \gamma T + \frac{1}{2}\lambda||x||^2$$

Expression	Meaning
$L(\theta)$	differentiable convex loss function which calculates the difference between the prediction value and the real value.
$\Omega(\theta)$	Regularized Term
T	Number of Leaves
γ	Learning rate

Table 1: Detals of the terms used in equations

Some of the most common convex loss functions that can be used are :

$$\text{Mean square loss function: } l(\hat{y_i}, y_i) = (\hat{y_i}, y_i)^2 2 \text{ and}$$
$$\text{Logistic loss function : } l(\hat{y_i}, y_i) = y_i ln(1 + e^{-\hat{y_i}}) + (1 - y_i) ln(1 + e^{\hat{y_i}})$$

In comparison to the gradient boosting algorithm , XGBoost increases the term $\frac{1}{2}\lambda||x||^2$. This increasing term can further prevent overfitting and even increase the generalization capability of our model.

However, the objective function in Eq(2) fails to optimize the traditional methods by existence of the model penalty items and functions as parameters. Hence, we use the following equation to predict the target y_i.

$$L(\theta) = \sum_{i=1}^{n} l(y_i\,, \hat{y_i}^{(t-1)} + S_t(T_i)) + \Omega(\theta) \tag{2}$$

The optimization goal here is to build a tree structure such that it minimizes our target function after each iteration. Every tree learns from the previous tree's conclusions and residuals.

$$residual = real\,value - predicted\,value$$

We can transform Eq(2) to Eq(3) through the two order Taylor Expansion.

$$L(\theta) = \sum_{i=1}^{n} \left[l(y_i\,, \hat{y_i}^{(t-1)}) + g_i S_t(T_i) + \frac{1}{2} h_i S_t^2(T_i) \right] + \Omega(\theta) \tag{3}$$

4 Environment setup and experiment

4.1 Experimental Setup

For all experiments , a Google colab notebook with following specifications was used :

GPU: 1xTesla K80 , 2496 CUDA cores , 12GB GDDR5 VRAM
CPU: 1xsingle core hyper threaded Xeon Processors @2.3Ghz
Disk space : 34 GB

The following programs as well as their dependencies were installed and imported into the notebook in order to run the experiment :

- NumPy
- Pandas
- Scikit-learn
- XGBoost

4.2 DataSet

Since we mainly focused on machine learning techniques to detect traditional and novel DDoS attack in SDN, we require training and testing network traffic to train and evaluate our classifier models. This training and testing network flow dataset should be robust.

4.2.1 Dataset Source

For our experiment , we have used the InSDN data set published in [3] paper. This is the most recent published dataset for Intrusion Detection in SDN. The data set was published in 2020.

The dataset is divided into three groups based on the traffic category and target machines. The first group contains attack traffic that targets the Mealsplotable-2 server, the second group contains attack traffic on the OVS machine and the last group contains normal traffic data only.

This InSDN dataset was generated using CICFlowMeter.

4.2.2 Dataset Characteristics

The dataset contains more than 80 features with 56 categories. The entire feature set is divided into 8 broader groups :

Network identifiers attributes	Network identifiers attributes IP address, Port number, protocol
Packet-based attributes	total number of packets in a forward and backward direction
Bytes-based attributes	total number bytes in the forward and backward direction
Interarrival time attributes	the interarrival time in both forward and backward directions
Flag attributes	SYN Flag, RST Flag, Push flag, etc.
Flow descriptors attributes	the number of packets and bytes in both forward and backward direction
Subflow descriptors attributes	the number of packet and bytes in forwarding and backward directions

Table 2: Dataset Characteristics

4.2.3 Data Set pre processing and feature Selection

From a 84 set of features a subset of 48 features is selected which are specific to SDN environment. We derived 6 key features from these 48 feature set. The feature mapping of the source dataset features and our derived features is shown below :

Source Dataset Feature	**Derived Dataset Feature**
Tot Fwd Pkts Tot Bwd Pkts	packet_count
Tot len Fwd Pkts Tot len Bwd Pkts	Byte_Count
Flow Pkts	Packet_rate
Fwd Pkts	Tx_rate
Bwd Pkts	Rx_rate
Flow Byte	Byte_rate

Table 3: Dataset Mapping

The rest of the features have been discarded as they either do not contribute to the detection of the attack or have high correlation with the already derived features.

There are 141953 data points in the reduced/derived dataset. We split the dataset into a train set and a test set of a 80-20 split ratio. We normalize the continuous and discrete values within the range of 0 to 1 using the following formula:

$$y_i^k = \frac{x_i^k - x_{min}^k}{x_{max}^k - x_{min}^k} \tag{4}$$

The table (no.) shows the training set and test set used in the experiment.

	Training Set	Testing Set	Total	Features
DDoS	58943	14586	73,529	6
Normal	54619	13805	68424	6

Table 4: DATA SETS USED IN EXPERIMENT

A flowchart showing the experimental procedure is shown below:

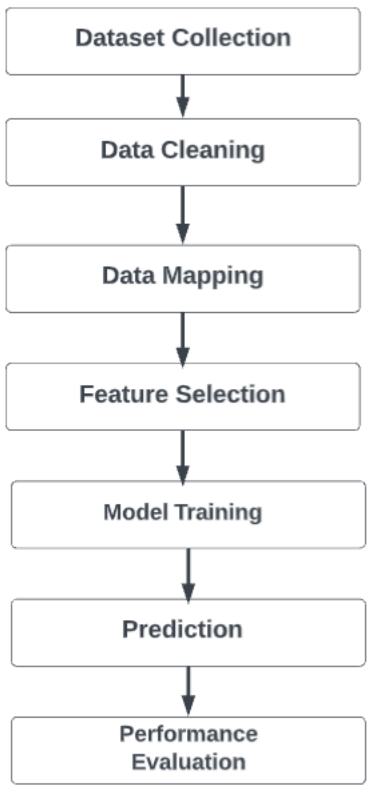


Figure 5: Flowchart showing experimental procedure

5 Result Analysis

In this section , we show the results of the classification using XGBoost algorithm classifier and then compare it to 4 other classifiers including KNN , Random Forest, Decision Tree and SVM. The performance measures used here are : accuracy , false alarm rate , Detection Rate and Execution time.

	Predicted DDoS	Predicted Normal	Total
Original DDoS	TP	FN	P
Original Normal	FP	TN	N

Table 5: CONFUSION MATRIX

The first three indicators are calculated using confusion matrix and the given formulas

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

Accuracy denotes the ratio of the number of samples that are correctly classified by the model to the total number of samples

$$FalseAlarmRate = \frac{FP}{FP + TN} \tag{6}$$

The False Alarm Rate(FAR) denotes the ratio of the number of samples that are incorrectly classified as attack traffic even though it is normal traffic to the number of all normal traffic samples

$$DetectionRate = \frac{TP}{TP + FN} \tag{7}$$

The detection rate(DR) denotes the ratio of the number of attack samples that are correctly classified in comparison to the total number of DDoS attack samples .

A table showing the performance metric of the classifiers is shown below :

	Accuracy (%)	**Detection Rate(%)**	**False Alarm Rate(%)**	**Execution Time (in sec)**
XGBoost	99.83	99.97	0.185	5.05
KNN	99.80	99.79	0.192	17
Random Forest	99.78	99.78	0.226	6.18
Decision Tree	99.42	99.71	0.857	11
SVM	96.59	99.97	6.615	1560

Table 6: Comparative Analysis between the classifiers

5.1 Accuracy

As we can see in Figure(6) , XGBoost classifier achieves highest accuracy compared to the other four classifiers. The Table no.6 shows that the accuracy of the XGBoost classifier is about 99.83% whereas that of KNN, RF,DT and SVM are 99.80%, 99.78%, 99.42% and 96.59% respectively. Even though all the classifiers have given quite a good accuracy score , XGBoost has outperformed the others in accuracy.

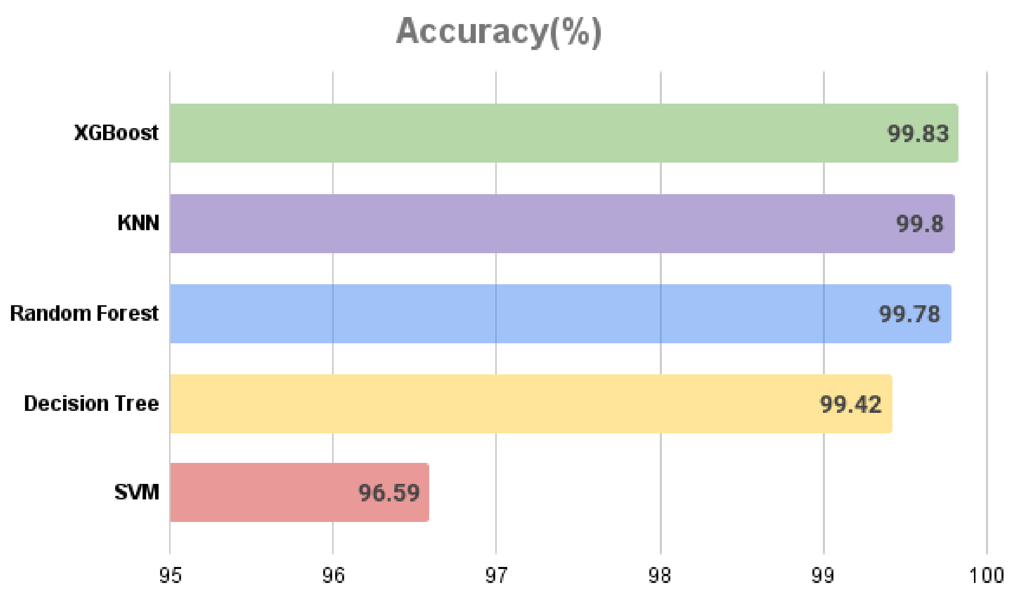


Figure 6: Accuracy

5.2 False Alarm Rate

Again in case of False Alarm Rate, XGBoost achieved a rate of 0.1851% which is also the lowest amongst all other classifiers used in this experiment. Table no. 6 shows details of the False alarm rates for the rest of the classifiers.We can see that KNN also has a very low False Alarm Rate of 0.192% . However SVM has a significant FAR score of 6.615% .

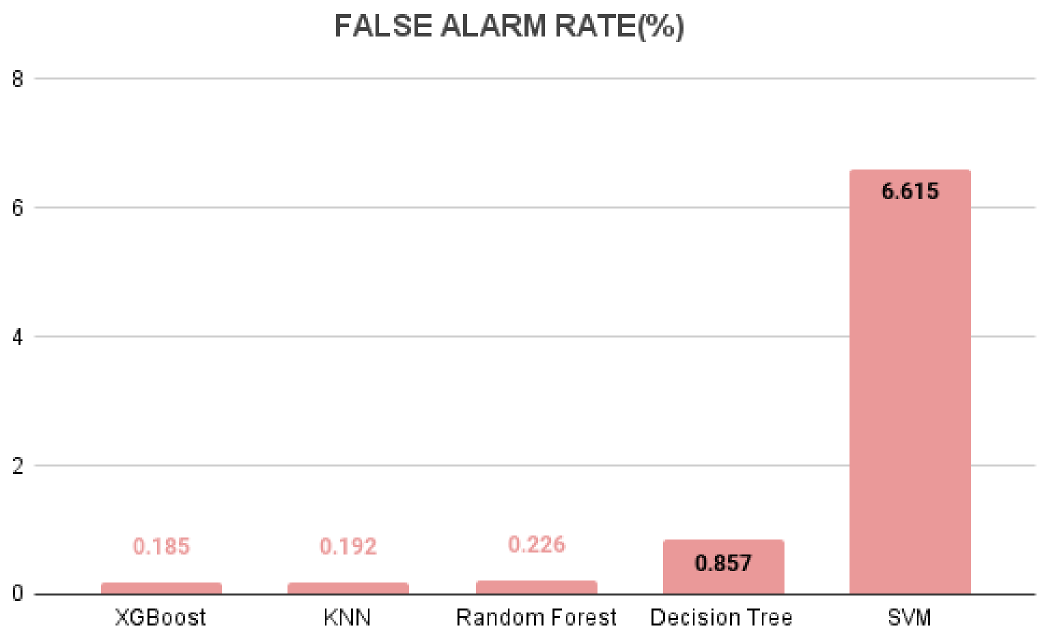


Figure 7: False Alarm Rate

5.3 Detection Rate

From Figure(8) , we can see that XGBoost classifier also has the highest detection rate of 99.97%. However, SVM classifier also has the same DR for the given dataset. From Table no.6 we can speculate that the other three classifiers have about same detection rate which is lower than XGBoost and SVM.

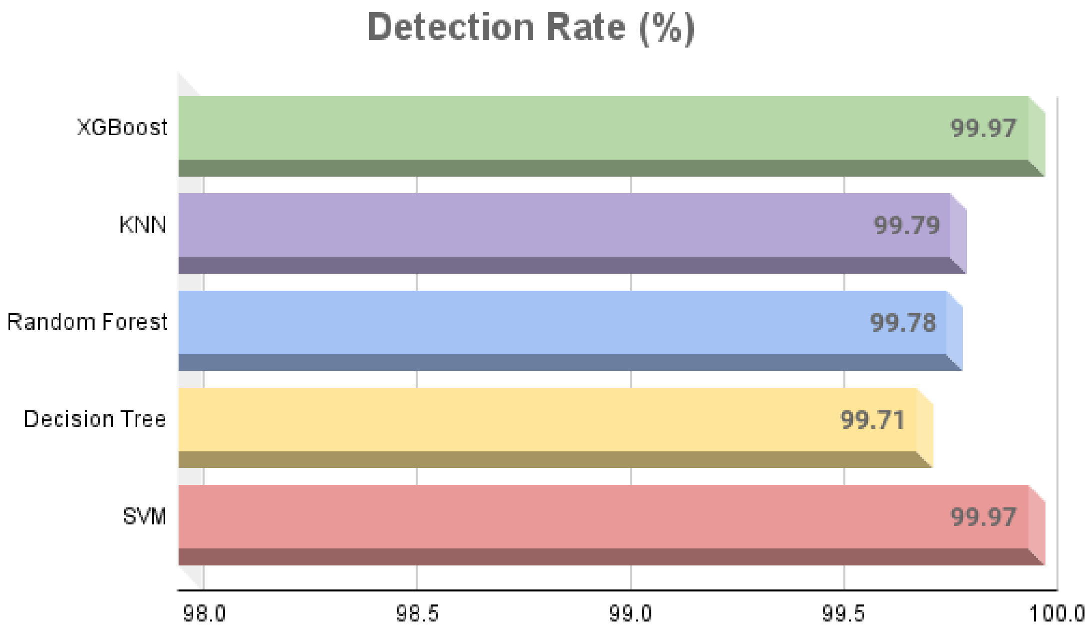


Figure 8: Detection Rate

5.4 Execution Time

In Table no.6 we can see that the execution time of XGBoost classifier is 5.03 seconds while that of KNN , Random Forest, Decision Tree and SVM are 17s,6.18s,11s and 1560s respectively. Figure(9) shows a logarithmic plot of the execution time of the different algorithms. We can see that XGBoost has the lowest run time and SVM has the highest run-time amongst the classifier algorithms.

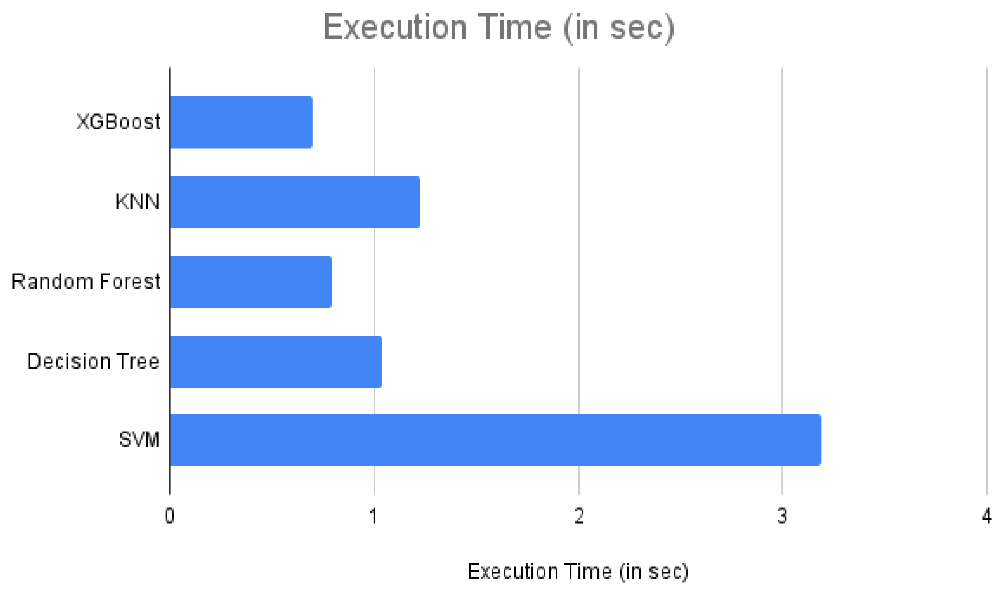


Figure 9: Logarithmic Plot for Execution Time

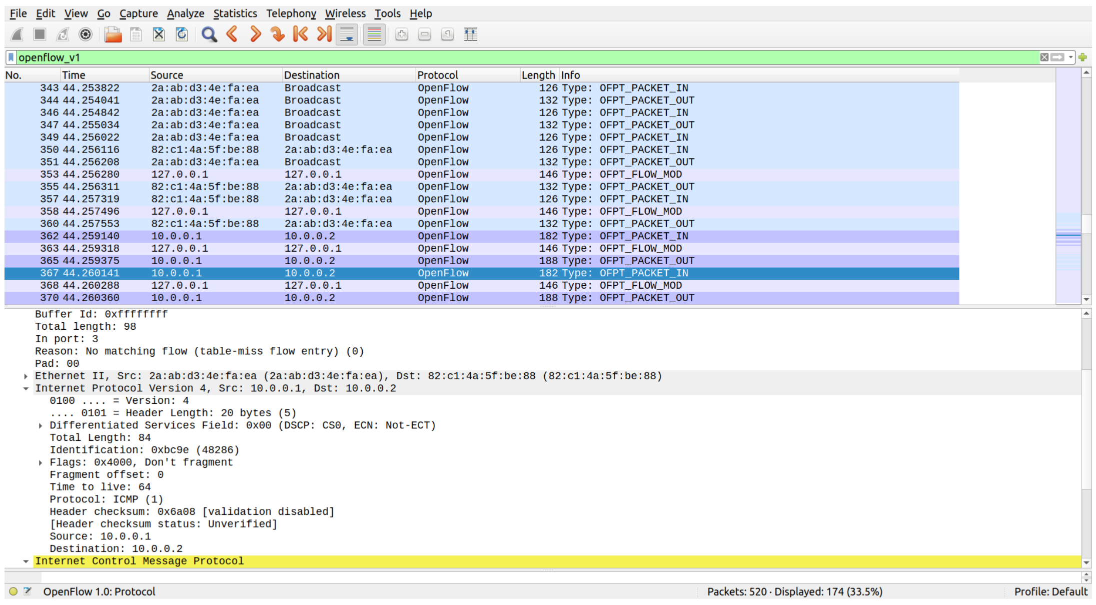
5.5 Summary

In summary , the results of binary classification on the test set denotes that XGBoost classifier outperform the rest classifiers in accuracy, detection rate , execution time and false alarm rate. Even though all classifiers have a very good performance and accuracy score , still one of the main concerns of our work was early detection of the DDoS attacks. As the experimental results prove, XGBoost classifier has a very low execution time with highest accuracy.So, we can say that our model is the best fit to fulfill our requirements and objectives.

6 Conclusion

Even though InSDN is a SDN specific anomaly detection dataset, it doesn't take into consideration the new type of DDoS attack mentioned in [2,3]. In future we plan to included new type of DDoS traffic in InSDN dataset and evaluate the performance of Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN) and Extreme Gradient Boosting (XGBoost).

We have already written a python script using scapy library to generate the new type of DDoS attack in the virtual SDN environment. The topology proposed above is created using Mininet, an emulator to emulate SDN topologies. Then we used packet capturing and analyzer like Wireshark and tshark to get the pcap file. This pcap file can be converted to csv format using either Wireshark or tshark. Tshark is the CLI version of Wireshark. Traffic captured by Wireshark has 242,000 features in total over 3000 protocols. We can collect SDN specific feature from pcap file using CICFlowMter [2].A detailed description about our pcap file containing new DDoS attack is mentioned below:


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
openflow_v1

```
No.    Time       Source             Destination        Protocol  Length Info
   343 44.253822  2a:ab:d3:4e:fa:ea  Broadcast          OpenFlow     126 Type: OFPT_PACKET_IN
   344 44.254041  2a:ab:d3:4e:fa:ea  Broadcast          OpenFlow     132 Type: OFPT_PACKET_OUT
   346 44.254842  2a:ab:d3:4e:fa:ea  Broadcast          OpenFlow     126 Type: OFPT_PACKET_IN
   347 44.255034  2a:ab:d3:4e:fa:ea  Broadcast          OpenFlow     132 Type: OFPT_PACKET_OUT
   349 44.256022  2a:ab:d3:4e:fa:ea  Broadcast          OpenFlow     126 Type: OFPT_PACKET_IN
   350 44.256116  82:c1:4a:5f:be:88  2a:ab:d3:4e:fa:ea  OpenFlow     126 Type: OFPT_PACKET_IN
   351 44.256208  2a:ab:d3:4e:fa:ea  Broadcast          OpenFlow     132 Type: OFPT_PACKET_OUT
   353 44.256280  127.0.0.1          127.0.0.1          OpenFlow     146 Type: OFPT_FLOW_MOD
   355 44.256311  82:c1:4a:5f:be:88  2a:ab:d3:4e:fa:ea  OpenFlow     132 Type: OFPT_PACKET_OUT
   357 44.257319  82:c1:4a:5f:be:88  2a:ab:d3:4e:fa:ea  OpenFlow     126 Type: OFPT_PACKET_IN
   358 44.257496  127.0.0.1          127.0.0.1          OpenFlow     146 Type: OFPT_FLOW_MOD
   360 44.257553  82:c1:4a:5f:be:88  2a:ab:d3:4e:fa:ea  OpenFlow     132 Type: OFPT_PACKET_OUT
   362 44.259140  10.0.0.1           10.0.0.2           OpenFlow     182 Type: OFPT_PACKET_IN
   363 44.259318  127.0.0.1          127.0.0.1          OpenFlow     146 Type: OFPT_FLOW_MOD
   365 44.259375  10.0.0.1           10.0.0.2           OpenFlow     188 Type: OFPT_PACKET_OUT
   367 44.260141  10.0.0.1           10.0.0.2           OpenFlow     182 Type: OFPT_PACKET_IN
   368 44.260288  127.0.0.1          127.0.0.1          OpenFlow     146 Type: OFPT_FLOW_MOD
   370 44.260360  10.0.0.1           10.0.0.2           OpenFlow     188 Type: OFPT_PACKET_OUT
```

```
    Buffer Id: 0xffffffff
    Total length: 98
    In port: 3
    Reason: No matching flow (table-miss flow entry) (0)
    Pad: 00
  ▸ Ethernet II, Src: 2a:ab:d3:4e:fa:ea (2a:ab:d3:4e:fa:ea), Dst: 82:c1:4a:5f:be:88 (82:c1:4a:5f:be:88)
  ▾ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 84
      Identification: 0xbc9e (48286)
    ▸ Flags: 0x4000, Don't fragment
      Fragment offset: 0
      Time to live: 64
      Protocol: ICMP (1)
      Header checksum: 0x6a08 [validation disabled]
      [Header checksum status: Unverified]
      Source: 10.0.0.1
      Destination: 10.0.0.2
  ▾ Internet Control Message Protocol
```

OpenFlow 1.0: Protocol   Packets: 520 · Displayed: 174 (33.5%)   Profile: Default

Figure 10: FlowDump in WireShark for New DDoS Attack

In future, after evaluating novel SDN specific dataset containing new DDoS attacks, we will save the best DDoS classifier as a pkl file and load it in the controller. The controller can request flow information from switches and predict whether the traffic is benign or malignant. If the traffic turns out to be malignant, necessary measures can be taken by the controller depending on the rules/policy put in action by the network administrator. For example, as controller can act as a hub, load-balancer and firewall application, we plan to restrict packets from malicious sources using firewall application if the model predicts PACKET_IN request from switch to controller as malignant traffic.

SDN security is now a major concern. DDoS attacks are becoming more and more prominent. In this report, we have shown that various work done on early detection of DDoS attacks, the dataset used in IDS training/testing and the scope to improve the solution to the problem. Through our research, we focused on new type of DDoS and novel SDN specific dataset to evaluate the performance of existing machine learning models.

References

[1] Zhuo Chen, Fu Jiang, Yijun Cheng, Xin Gu, Weirong Liu, and Jun Peng. Xgboost classifier for ddos attack detection and analysis in sdn-based cloud. In *2018 IEEE international conference on big data and smart computing (bigcomp)*, pages 251–256. IEEE, 2018.

[2] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of encrypted and vpn traffic using time-related. In *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pages 407–414. sn, 2016.

[3] Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D Jurcut. Insdn: A novel sdn intrusion dataset. *IEEE Access*, 8:165263–165284, 2020.

[4] Heena Kousar, Mohammed Moin Mulla, Pooja Shettar, and DG Narayan. Detection of ddos attacks in software defined network using decision tree. In *2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, pages 783–788. IEEE, 2021.

[5] Seyed Mohammad Mousavi and Marc St-Hilaire. Early detection of ddos attacks against sdn controllers. In *2015 international conference on computing, networking and communications (ICNC)*, pages 77–81. IEEE, 2015.

[6] Huseyin Polat, Onur Polat, and Aydin Cetin. Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models. *Sustainability*, 12(3):1035, 2020.

[7] Kshira Sagar Sahoo, Bata Krishna Tripathy, Kshirasagar Naik, Somula Ramasubbareddy, Balamurugan Balusamy, Manju Khari, and Daniel Burgos. An evolutionary svm model for ddos attack detection in software defined networks. *IEEE Access*, 8:132502–132513, 2020.

[8] Reza Bakhtiari Shohani and Seyed Akbar Mostafavi. Introducing a new linear regression based method for early ddos attack detection in sdn. In *2020 6th*

International Conference on Web Research (ICWR), pages 126–132. IEEE, 2020.

[9] Junyuan Tan, Shan Jing, Lei Guo, and Bin Xiao. Ddos detection method based on gini impurity and random forest in sdn environment. In *2021 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pages 601–606. IEEE, 2021.

[10] Liang Tan, Yue Pan, Jing Wu, Jianguo Zhou, Hao Jiang, and Yuchuan Deng. A new framework for ddos attack detection and defense in sdn environment. *IEEE Access*, 8:161908–161919, 2020.

[11] Di Wu, Jie Li, Sajal K Das, Jinsong Wu, Yusheng Ji, and Zhetao Li. A novel distributed denial-of-service attack detection scheme for software defined networking environments. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[12] Jin Ye, Xiangyang Cheng, Jian Zhu, Luting Feng, and Ling Song. A ddos attack detection method based on svm in software defined network. *Security and Communication Networks*, 2018, 2018.